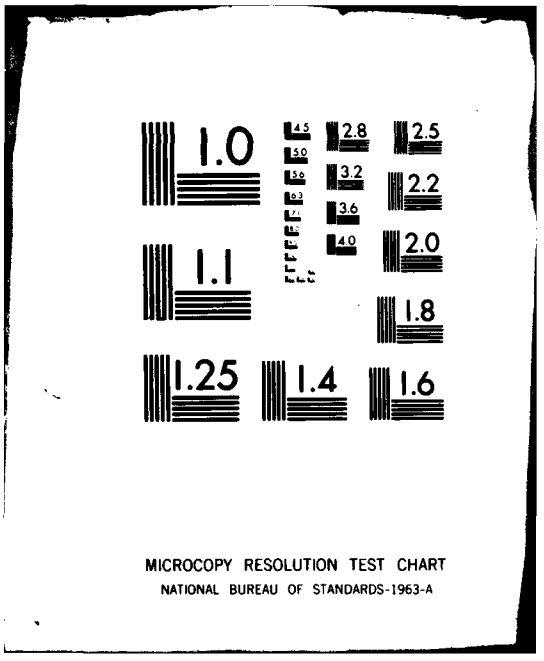
AD-A083 727 MICHIGAN STATE UNIV EAST LANSING DEPT OF COMPUTER SCIENCE F/G 9/2
A STUDY OF A PROCEDURE FOR REDUCING THE FEATURE SET OF WORKLOAD--ETC(U)
FEB 80 H D HUGHES AFOSR-78-3547

UNCLASSIFIED AFOSR-TR-80-0259 NL

END
DATE FILMED
4-80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AFOSR-TR- 80-0259

A Study of a Procedure for Reducing the Feature Set of Workload Data*

Herman D. Hughes
Department of Computer Science
Michigan State University
East Lansing, Michigan 48824

ABSTRACT--Workload models are extremely important for computer performance evaluation. The problem of feature reduction for the purpose of the formulation of workload models has received widespread attention. This paper briefly reviews existing schemes for feature selection and reduction, and proposes a feature reduction algorithm based on a linear decision-tree classifier. An example is presented to illustrate the use and validity of this algorithm.

KEY WORDS--Density function, clustering, workload data, feature reduction, linear machine, discriminant function, decision tree, classifier, misclassifications, partition.

*Research supported by AFOSR Grant 78-3547

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR-TR-80-0259	2. GOVT ACCESSION NO.: AD-A083727
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): A STUDY OF A PROCEDURE FOR REDUCING THE FEATURE SET OF WORKLOAD DATA	5. TYPE OF REPORT & PERIOD COVERED: Final rept
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Herman D. Hughes	8. CONTRACT OR GRANT NUMBER(s): AFOSR 78-3547
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Michigan State University, College of Engineering - Dept of Computer Science, East Lansing, MI 48824	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research/NM, Bolling AFB, Washington, DC 20332	12. REPORT DATE: February 1980 13. NUMBER OF PAGES: 19
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):	15. SECURITY CLASS. (of this report): UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

density function, clustering, workload data, feature reduction, linear machine, discriminant function, decision tree, classifier, misclassifications, partition

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Workload models are extremely important for computer performance evaluation. The problem of feature reduction for the purpose of the formulation of workload models has received widespread attention. This paper briefly reviews existing schemes for feature selection and reduction, and proposes a feature reduction algorithm based on a linear decision-tree classifier. An example is presented to illustrate the use and validity of this algorithm.

I. Introduction

It is a well-established fact that the workload characterization is a very important step in the performance evaluation of a computer system [1,4]. The workload of a computer system can be roughly defined as the set of all inputs (programs, data, commands) the system receives from its environment [4]. In forming the workload model, many measurements can be made on this set of inputs through accounting logs, software or hardware monitors. Among the most frequently used variables, for example, are CPU time, core requirements, number of disk/drum requests, the length of I/O requests, the number of tape I/Os, etc.

Quite often, the data collected is too voluminous to be used directly. A procedure is required to extract or select a subset of the data that can still effectively characterize the workload. Hence, feature selection is an essential procedure for building a workload model for at least four major reasons:

(i) Some of the monitoring activities are time-consuming and costly. By reducing the feature set, unnecessary monitoring may be eliminated.

(ii) Storage requirements are reduced, both in the data-collecting phase and in the workload formulation and testing phases.

(iii) Computational cost is reduced in the designing and the testing of workload models.

(iv) Reducing the number of features will reduce the complexity of the model.

This report describes an algorithm for reducing the number of features by way of a linear decision-tree classifier which at worst case, performs as well as the linear machine frequently used in pattern recognition.

II. Background of Feature Selection

The problem of feature selection has received much attention in the field of Pattern Recognition. Fu, Min and Li [5] summarized the feature selection methods into four categories:

(i) Information theoretic approach.

This approach assumes that the data has a multivariate Gaussian density, and uses the divergence between two classes or a general separability measure as a criterion for feature selection.

(ii) Direct estimation of error probability.

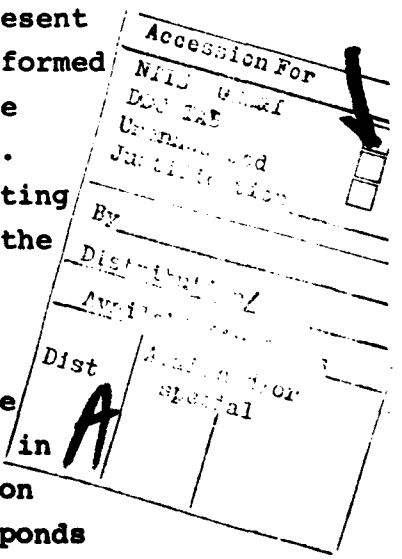
The underlying assumption for this method is that the distribution density of the data is not known, so small windows (Parzen-Window) [10] in the sample space are used to estimate the probability density. The probability of misclassification is used as the criterion for reducing the number of features.

(iii) Feature-space transformation.

The Karhunen-Loève (K-L) expansion is used here, which involves finding the eigenvalues and the corresponding eigenvectors of the covariance matrix of the sample data [6]. The eigenvectors represent the orthogonal coordinates (axes) in the transformed space, and the corresponding eigenvalues may be seen as the variances with respect to the axes. The feature reduction is then achieved by deleting the axis in the new space that corresponds to the smallest eigenvalue.

(iv) Stochastic automata approach.

A learning automata is constructed as a feature selection scheme where the automata is defined in terms of training samples and a certain decision rule. Each action made by the automata corresponds to choosing a certain subset of the feature set.



The automata receives one penalty when an incorrect action is taken. By minimizing the total number of penalties, and then identifying the action that is most frequently taken by the automata, we obtain the best feature subset.

While good results have been obtained from many of the feature reduction techniques previously mentioned (especially for problems in pattern recognition), much valuable information which is essential for workload modelling is lost in the reduction process.

Agrawala and Mohr [2] discussed the feasibility of using the methods in pattern recognition for the workload characterization problem, and concluded that the feature reduction by direct re-clustering does not work well for workload characterization.

Mamrak and Amar [9] used a backward regression method in which the probability of error of a particular feature subset for describing the point densities is used as the criterion for eliminating features. This procedure produces good results only when the data used contains some features which are relatively insignificant.

The eigenvalue analysis procedure was the first technique used for this research. However, the results were unsatisfactory for the following reasons: (1) a reduced feature space is produced where each new feature is a linear combination of the original features, and (2) the new feature space is not optimized for identifying clusters, but for retaining the variance of the original space.

Hierarchical clustering [8] and multivariate regression were used to determine if some relationship exists between features, and then eliminate features by substitution. But these techniques only work when there exists a high correlation between features, and this was not the case for the workload data considered here.

In the next section, a procedure for reducing the feature set for workload data is presented. This procedure relies on some basic concepts of pattern recognition.

III. Feature Reduction by Decision-Tree

The two problems most frequently encountered in feature reduction are:

1) The underlying distribution density for each class is seldom known.
2) The selected subset of features produce intolerable error rate when reclassifying the sample data.

In the case where the underlying distribution densities are not known, it is sometimes desirable to find out if the classes of samples are linearly separable, (i.e., if there exists a set of linear discriminant functions that can separate all classes in the sample space).

Consider the case where there are m classes $C_1,\ldots,C_m$ in the sample set, and let $X^t = (x_1 \ldots x_p)$ denote a feature vector, where $x_1 \ldots x_p$ are the p feature values. Then a linear discriminant function for a given class C_i is defined as:

$$g_i(X) = \bar{w}_i^{\,t} X + w_{io} \quad , \quad i = 1,\ldots,m \qquad (1)$$

where $\bar{w}_i^{\,t} = (w_{i1},\ldots,w_{ip})$ and w_{io} are weighting coefficients such that, if $X \varepsilon C_i$, then

$$g_i(X) > g_j(X) \text{ for every } j \neq i. \qquad (2)$$

The problem of classification now becomes one of finding the $\bar{w}_i$ and w_{io} in (1) such that (2) is satisfied. After finding each $\bar{w}_i$ and w_{io} for all $i = 1,\ldots,m$, the classification proceeds as follows: for any new data point $Z = (z_1,\ldots,z_p)$, assign Z to class C_k if $g_k(Z) > g_\ell(Z)$ for every $\ell \neq k$, and leave Z undecided if there are ties. This type of classifier is called the <u>linear machine</u> [3]. Figure 1 gives an example of a 3-class classification problem. The discriminant functions g_1, g_2 and g_3 are obtained from three training sets C_1, C_2 and C_3. Three regions R_1, R_2 and R_3 are formed based on the relationships between g_1, g_2 and g_3. Any new

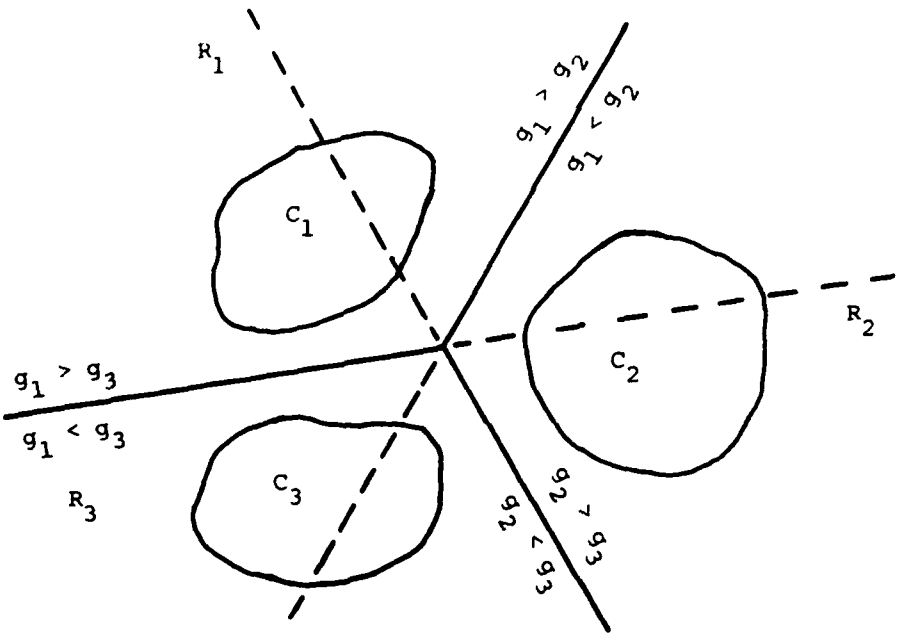


Figure 1. A 3-class problem in linear-machine.

data point Z that lies within R_i would then have $g_i(Z) > g_j(Z)$ for $j \neq i$, and hence classified as in class i. Note that the set of discriminant functions $\{g_1 \ldots g_m\}$ need not be unique.

This linear machine can be further generalized into quadratic or polynomial discriminant functions in order to obtain better fits to the boundaries. However, the number of coefficients and the computational complexity will increase exponentially, so the linear discriminant functions are preferred if the error rate is tolerable.

In view of the second problem, deleting features from the original feature set is equivalent to projecting the sample space into a lower dimensional subspace without any transformation, hence causing changes in the distance between patterns as well as classes. So, unless the classes are already well-separated in a particular subspace, feature reduction by brute-force is liable to introduce intolerable errors.

Instead of trying to find a subset of features to classify <u>all</u> classes in one step, we concentrate on just classifying

a few classes (e.g., one or two) from the rest at each stage of an iterative procedure. As a result of this procedure, all classes may eventually be classified.

For example, Figure 2 illustrates a decision-tree classifier: g_1 discriminates C_1 from the rest by letting

$$g_1(X) > 0 \quad \text{if} \quad X \in C_1$$
$$< 0 \quad \text{if} \quad X \notin C_1$$

Then g_2 separates C_2 and C_3 by letting

$$g_2(X) > 0 \quad \text{if} \quad X \in C_2$$
$$< 0 \quad \text{if} \quad X \notin C_2$$

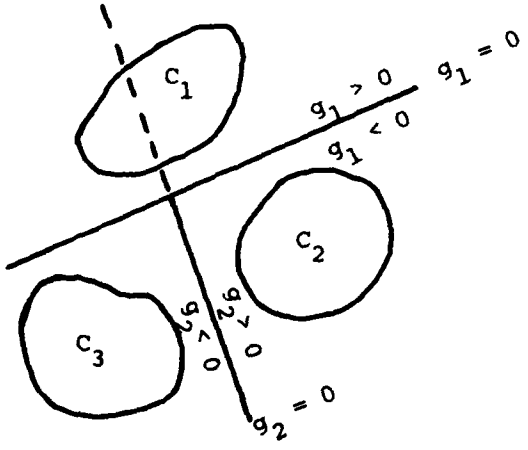
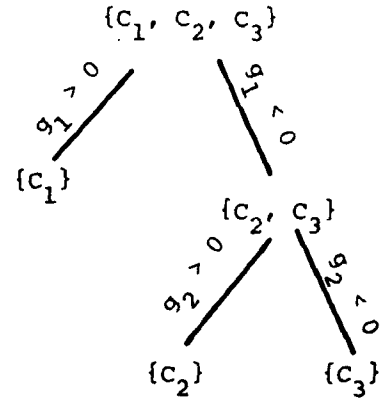


Figure 2. (a) A 3-class problem with 2 discrimant functions.



(b) A decision-tree to classify 3 classes C_1, C_2, and C_3.

Compared to the classifier in the previous example, we see that the number of discriminant functions is reduced. But more importantly, since g_1 and g_2 are only involved in separating one class from the others, the chance of successfully reducing features is considerably increased when the number of features is high.

IV. Algorithm for Formulating the Decision-Tree

A procedure for setting up the decision-tree classifier and hence achieving feature reduction is presented here.

Step 1. Obtain the class information of sample data in the scaled (normalized) space.

This procedure requires labeled data set in order to design and test the classifier. Since no absolute standard usually exists for labeling the data classes, a clustering procedure on the data set with all features used is often required.

Step 2. For each feature j in class C_i, find the mean M_{ij} and the standard deviation SD_{ij} in the normalized space.

Step 3. Use each feature to form a partition of the set of classes $C = \{C_1,\ldots,C_m\}$.

First, measure the "distance" of class C_i and C_k on feature j by defining

$$D_j(C_i,C_k) = \frac{|M_{ij} - M_{kj}|}{SD_{ij} + SD_{kj}}, \quad j = 1,..,m.$$

The classes C_i and C_k are called "close" to each other on feature j if

(i) $D_j(C_i,C_k) < t$ for some threshold t,

(ii) there exists a class C_e such that $D_j(C_i,C_\ell) < t$ and $D_j(C_\ell,C_k) < t$

(i.e., D is transitive). Now, form a partition of $C = \{C_1,\ldots C_m\}$ over feature j by putting the close classes into the same partition "——".

$$P_j = \{\overline{C_1\ldots C_u}\ \overline{C_v\ldots C_r},\ldots\}.$$

Step 4. Rank each feature by:

(i) the number of classes it can separate; then
(ii) the magnitude of distance $D(C_i, C_k)$ by which it separates the classes.

Step 5. By taking the intersections of different partitions $P_1, P_2 \ldots P_p$, we hope to find one or more combinations that completely partition C into single classes.

The intersection $P_{a \cap b}$ of two partitions P_a and P_b is defined as:

$$P_{a \cap b} = \{\overline{C_k C_\ell \ldots} \mid \overline{C_k C_\ell \ldots} \in P_a \text{ and } \overline{C_k C_\ell \ldots} \in P_b\}.$$

For example, if $P_1 = \{\overline{C_1 C_2 C_3}, \overline{C_4}\}$, $P_2 = \{\overline{C_1 C_3}, \overline{C_2 C_4}\}$, then $P_{1 \cap 2} = P_1 \cap P_2 = \{\overline{C_1 C_3}, \overline{C_2}, \overline{C_4}\}$.

The result of Step 4 is used here for selecting features. If the next ranking feature does not provide any additional refinement in the partition, then we may skip this feature.

Step 6. If some classes are not separable in the above step, then the linear machine mentioned in the previous section is used on the features that best separate these classes (from Steps 3 and 4) in order to derive a linear discriminant function.

Step 7. Form the decision-tree with the features or linear discriminant functions selected. The root node of the tree is the entire sample set. Its successors are the partitions by one of the features (or the linear discriminant functions) selected in step 5 or 6. Repeat this branching process until the leaves of the tree are all

single classes.

Step 8. Test the decision-tree classifier with the test data. If the error rate is greater than desired, go back to Step 5.

Note that there is no guarantee that the feature-reduction algorithm presented will always work; for some of the data may be truely non-feature-reducible. But viewing the linear-machine as a special case of a one-level decision-tree with all available features, this algorithm will do at least as good as the linear machine.

V. Experiment and Result

A sample data set with 1200 jobsteps were collected from a DEC system-10 at Wright-Patterson Air Force Base [7]. The 10 features of the workload are listed in Table 1.

As a first step, a clustering program was used to cluster these 1200 jobsteps with all 10 features. Eight clusters (classes) were produced that reasonably separated the 1200 data points. During the clustering, normalized feature values were used to avoid having any large-scale feature(s) dominating the clustering. This normalized scale was used throughout this experiment. The centers of these 8 classes in scaled space are listed in Table 2.

In Steps 2 to 4, the first 600 jobsteps were used to design the classifier. The partitions of the set of all classes $C = \{C_1,\ldots,C_8\}$ are listed in Table 3 along with the feature ranking. P_1 is the partition produced by feature 1, P_2 is the partition produced by feature 2, etc.

In Step 5, there are several ways of obtaining a partition with all classes separated. For example:

$$P_1 \cap P_8 = \{\overline{C_1C_2C_5C_8},\ \overline{C_3},\ \overline{C_4},\ \overline{C_6},\ \overline{C_7}\}$$

Since the next ranking partition (P_{10}) does not provide an additional refinement, we proceed to select the next highest ranking partition (P_2).

$$P_{1\cap 8}\cap P_2 = \{\overline{C_1C_5C_8},\ \overline{C_2},\ \overline{C_3},\ \overline{C_4},\ \overline{C_6},\ \overline{C_7}\}$$

Again, skip partitions P_6, P_3 and P_5 for the same reason given for skipping P_{10}. Now, to continue this process we select P_4, and obtain:

$$P_{1\cap 8\cap 2}\cap P_4 = \{\overline{C_1C_8},\ \overline{C_2},\ \overline{C_3},\ \overline{C_4},\ \overline{C_5},\ \overline{C_6},\ \overline{C_7}\}$$

Finally, we choose P_9 which yields:

$$P_{1\cap 8\cap 2\cap 4}\cap P_9 = \{\overline{C_1},\ \overline{C_2},\ \overline{C_3},\ \overline{C_4},\ \overline{C_5},\ \overline{C_6},\ \overline{C_7},\ \overline{C_8}\}$$

Using different combinations, we also may have:

$$P_1\cap P_2\cap P_5\cap P_7\cap P_8 = \{\overline{C_1},\ \overline{C_2},\ \overline{C_3},\ \overline{C_4},\ \overline{C_5},\ \overline{C_6},\ \overline{C_7},\ \overline{C_8}\}$$

The decision-tree we obtained is depicted in Figure 3. Notice that the order of the feature appeared in the decision-tree may not necessarily correspond to the order of ranking. The discriminant functions, shown as the thresholds on the branches of the tree, were determined experimentally. For example, the first threshold (i.e., for feature #2) was set equal to 190 so as to minimize the error of classification between $\{C_2\}$ and $\{C_1, C_3, C_4, C_5, C_6, C_7, C_8\}$. The set of features is reduced from 10 to 6 features, namely, 1, 2, 4, 5, 7, and 8. The 600 design samples and the second 600 test samples are re-classified according to this classifier, and the results are listed in Table 4. The confusion matrix of re-classification on the total 1200 samples is shown in Table 5. After observing the data in Tables 4 and 5, it is obvious that the feature reduction procedure presented in this paper produced impressive results when applied to the workload data under consideration.

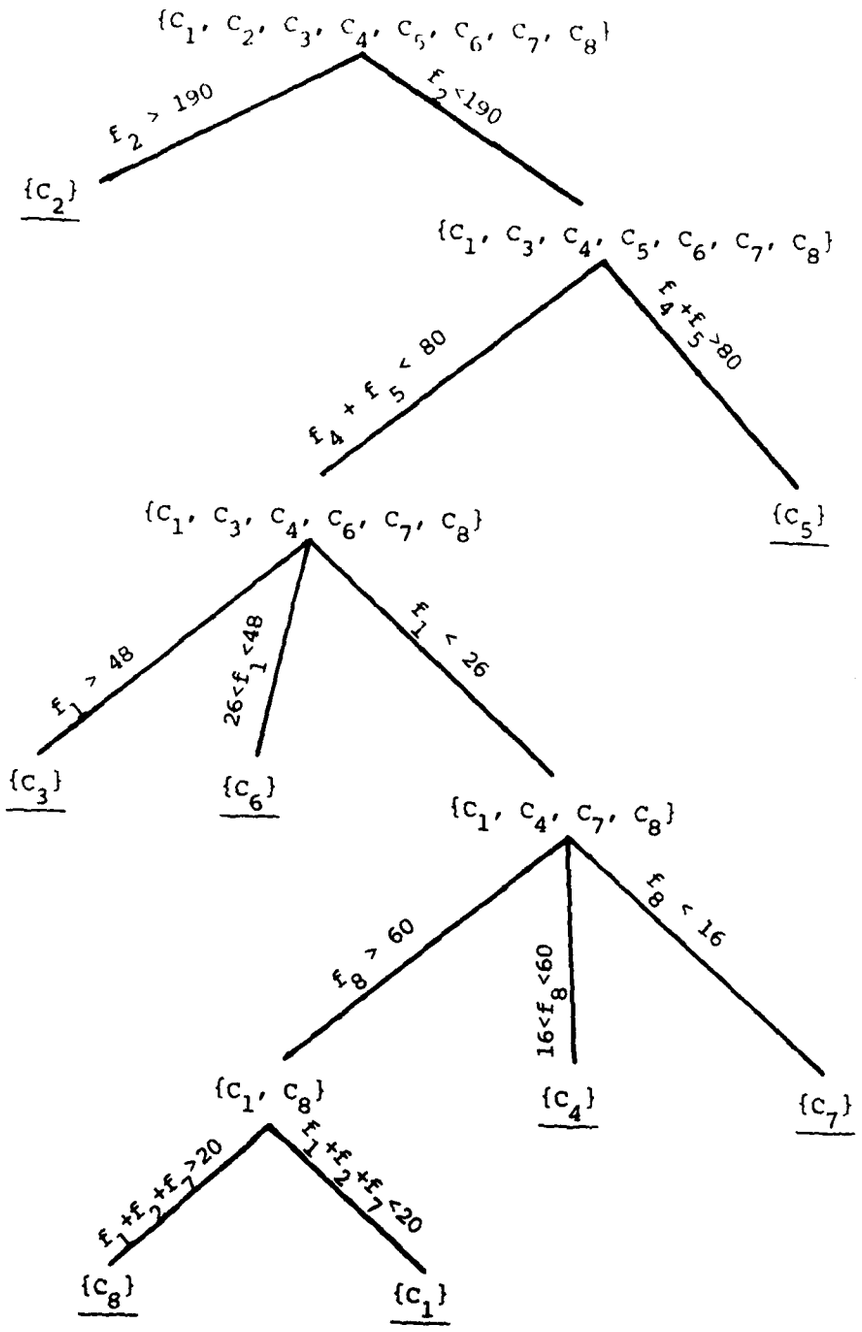


Figure 3. The decision-tree classifier. C_i denotes class i and f_k denotes the value of feature k.

VI. Summary and Conclusion

The formulation of workload model is an essential requirement for performance evaluation of a computer system. Often the measurements made on the workload data through software or hardware monitors can be quite voluminous. In order to reduce the features and yet preserve as much information in classifying the workload classes, we have proposed a decision-tree-classifier feature reduction algorithm. This algorithm can be characterized as a linear machine in a sequential decision-tree, in that each decision rule involves only classifying one or a few classes at a time using a linear combination of a subset of features. Viewing the linear machine as a one-level all-feature decision-tree, we can see that this algorithm will do at least as well as the linear machine.

Many feature ranking and selection algorithms have been proposed, but here we use a rather straightforward method to simplify the procedures involved.

We have reduced the set of features from 10 to 6 with a total error rate of 3.08%. If one more feature (i.e., feature 4) were deleted, then the error rate would be roughly doubled, but still within 7%. Also, the decision-tree may not be the optimal in that we did not exhaust all the combinations of the partitions by features. Hence, this algorithm does have the potential for achieving feature reduction with a tolerable rate of error.

ACKNOWLEDGEMENTS

I would like to express my appreciation for two graduate students (Liang Li and Jeff Perdue) who provided a significant contribution to the development of this paper.

References

1. A. K. Agrawala, J. M. Mohr, and R. M. Bryant, "An Approach to the Workload Characterization Problem," Computer, June 1976, pp. 18-32.

2. A. K. Agrawala, J. M. Mohr, "The Relationship Between the Pattern Recognition Problem and the Workload Characterization Problem," Technical Report, University of Maryland, May 1977.

3. R. D. Duda and P. E. Hart, "Pattern Classification and Scene Analysis," New York: Wiley, 1973, Ch. 5.

4. D. Ferrari, "Computer Systems Performance Evaluation," Englewood Cliff: Prentice-Hall, 1978, Ch. 5.

5. K. S. Fu, P. J. Min, and T. J. Li, "Feature Selection in Pattern Recognition," IEEE Trans. on Sys. Sci. and Cyber., Vol. SSC-6, No. 1, January 1970, pp. 33-39.

6. K. Fukunaga and D. R. Olsen, "An Algorithm for Finding Intrinsic Dimensionality of Data," IEEE Trans. Comp., Vol. C-20, pp. 176-183, Feb. 1971.

7. H. D. Hughes, "Workload Characterization of Computer Systems," IX Conference Proceedings, The Computer Measurement Group, 1978, pp. 81-93.

8. A. K. Jain and R. Dubes, "Feature Definition in Pattern Recognition with Small Sample Size," Pattern Recognition, Vol. 10, 1978, pp. 85-97.

9. A. Mamrak and P. D. Amer, "A Feature Selection Tool for Workload Characterization," Technical Report, Ohio University, 1976.

10. E. Parzen, "On the Estimation of a Probability Density Function and Mode," Ann. Math. Stat., Vol. 33, pp. 1065-1076, 1962.

Feature	Name
1	CPU time (sec)
2	Core size (K- words)
3	Number of disk I/O's
4	Teletype I/O (number of characters)
5	Number of interaction counts
6	Number of tape I/O counts
7	Number of core increases
8	Average core increase
9	Number of core decreases
10	Average core decrease

Table 1 The Features

CLUSTER	f_1	f_2	f_3	f_4	f_5
1	-1.6756	-1.0978	-1.4418	-.6985	-.7196
2	-2.3260	305.0797	-2.1604	-3.1165	-1.8684
3	79.5851	13.5612	86.2258	.5887	.1518
4	6.0211	8.5139	7.3398	4.5257	3.7703
5	5.3334	-.5153	.4891	75.4322	79.6747
6	41.0921	9.3341	.2677	1.2483	-1.8684
7	1.3627	18.8346	5.7694	1.3184	.6205
8	9.9732	7.9743	5.9525	-.9423	-.7925

CLUSTER	f_6	f_7	f_8	f_9	f_{10}
1	-.4985	-1.7256	-1.2986	-1.6697	-1.7356
2	-.5330	-4.1164	-3.8799	-4.1962	-3.1861
3	-.5330	62.0908	5.1289	35.7363	30.6155
4	-.5330	14.6205	20.6923	-.2509	82.4214
5	-.5330	1.2942	4.3514	-2.9503	-3.0969
6	199.0540	4.5673	36.0107	-2.1198	-3.1861
7	-.5330	11.3585	93.3493	-1.8890	.0841
8	-.5330	14.5811	2.9368	25.5050	12.9688

Table 2. Cluster Centers in Scaled Space

Partition	Rank
$P_1 = \{\overline{C_1C_2C_4C_5C_7C_8},\ \overline{C_3},\ \overline{C_6}\}$	1
$P_2 = \{\overline{C_2},\ \overline{C_1C_3C_4C_5C_6C_7C_8}\}$	4
$P_3 = \{\overline{C_1C_2C_4C_5C_6C_7C_8},\ \overline{C_3}\}$	6
$P_4 = \{\overline{C_1C_2C_3C_4C_6C_7C_8},\ \overline{C_5}\}$	8
$P_5 = \{\overline{C_1C_2C_3C_4C_6C_7C_8},\ \overline{C_5}\}$	7
$P_6 = \{\overline{C_1C_2C_3C_4C_5C_7C_8},\ \overline{C_6}\}$	5
$P_7 = \{\overline{C_1C_2C_4C_5C_6C_7C_8},\ \overline{C_3}\}$	9
$P_8 = \{\overline{C_1C_2C_3C_5C_8}, \overline{C_4C_6}, \overline{C_7}\}$	2
$P_9 = \{\overline{C_1C_2C_4C_5C_6C_7},\ \overline{C_3C_8}\}$	10
$P_{10} = \{\overline{C_1C_2C_5C_6C_7C_8},\ \overline{C_3}, \overline{C_4}\}$	3

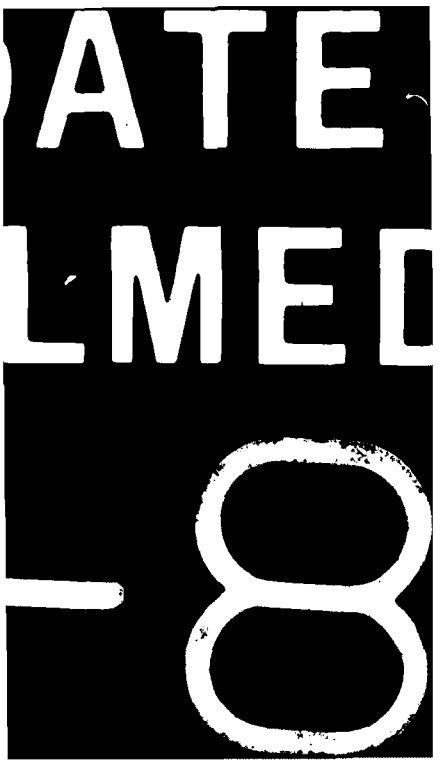
Table 3. Partitions by features

Cluster	Original No.	Correct Class.	Misclassification Type I	Type II
1	548	538	10	8
2	0	0	0	0
3	3	3	0	1
4	7	5	2	5
5	7	7	0	0
6	3	3	0	1
7	3	3	0	0
8	29	21	8	5
Total	600	580	20	20

Table 4(a) Result of the classifier with design set (err = 3.33%)

Cluster	Original No.	Correct Class.	Misclassification Type I	Type II
1	550	548	2	9
2	1	1	0	0
3	10	10	0	1
4	3	0	3	3
5	3	3	0	1
6	0	0	0	2
7	6	6	0	0
8	27	15	12	1
Total	600	583	17	17

Table 4(b) Result of the classifier with test set (err = 2.83%)


ATE
LMED
8